THINK JAVA CLASS EXERCISES

Instructor: Chia James Chang
Institution: TSG of ROLF
Date and place: 10/16, 10/23, 10/30, 11/6, 11/13 and 11/20 total 6 meets on Tuesday nights, from 7:30 to 9:40 pm at E6 in ROLCC

Extracted from the book of “Think Java: How to Think Like a Computer Scientist” By Allen B. Downey

Table of Contents

Chapter 1 The way of the program

Exercise 1

```
class Hello {
    // main: generate some simple output
    public static void main(String[] args) {
        System.out.println("Hello, world.");
    }
}
```

Before you do anything else, find out how to compile and run a Java program in your environment. Some environments provide sample programs similar to the example in Section 1.5.

1. Type in the “Hello, world” program, then compile and run it.
2. Add a print statement that prints a second message after the “Hello, world!”. Something witty like, “How are you?” Compile and run the program again.
3. Add a comment to the program (anywhere), recompile, and run it again. The new comment should not affect the result.

This exercise may seem trivial, but it is the starting place for many of the programs we will work with. To debug with confidence, you have to have confidence in your programming environment. In some environments, it is easy to lose track of which program is executing, and you might find yourself trying to debug one program while you are accidentally running another. Adding (and changing) print statements is a simple way to be sure that the program you are looking at is the program you are running.

Exercise 2

It is a good idea to commit as many errors as you can think of, so that you see what error messages the compiler produces. Sometimes the compiler tells you exactly what is wrong, and all you have to do is fix it. But sometimes the error messages are misleading. You will develop a sense for when you can trust the compiler and when you have to figure things out yourself.

1. Remove one of the open squiggly-braces.
2. Remove one of the close squiggly-braces.
3. Instead of main, write mian.
4. Remove the word static.
5. Remove the word public.
6. Remove the word System.
7. Replace println with Println.
8. Replace println with print. This one is tricky because it is a logic error, not a syntax error. The statement System.out.print is legal, but it may or may not do what you expect.
9. Delete one of the parentheses. Add an extra one.

Chapter 2 Variables and types

Exercise 1

If you are using this book in a class, you might enjoy this exercise: find a partner and play "Stump the Chump”: Start with a program that compiles and runs correctly. One player turns away while the other player adds an error to the program. Then the first player tries to find and fix the error. You get two points if you find the error without compiling the program, one point if you find it using the compiler, and your opponent gets a point if you don’t find it.

Exercise 2

```
class Hello {
    public static void main(String[] args) {
        String firstLine;
        firstLine = "Hello, again!";
        System.out.println(firstLine);
    }
}
```

1. Create a new program named Date.java. Copy or type in something like the “Hello, World” program and make sure you can compile and run it.
2. Following the example in Section 2.4, write a program that creates variables named day, date, month and year. day will contain the day of the week and date will contain the day of the month. What type is each variable? Assign values to those variables that represent today’s date.

3. Print the value of each variable on a line by itself. This is an intermediate step that is useful for checking that everything is working so far.
4. Modify the program so that it prints the date in standard American form: Saturday, July 16, 2011.
5. Modify the program again so that the total output is:

```
American format:
Saturday, July 16, 2011
European format:
Saturday 16 July, 2011
```

The point of this exercise is to use string concatenation to display values with different types (int and String), and to practice developing programs gradually by adding a few statements at a time.

Chapter 3 Void methods

Exercise 1

Draw a stack frame that shows the state of the program in Section 3.10 when main invokes printTime with the arguments 11 and 59.

```
Section 3.10
public static void printTime(int hour, int minute) {
    System.out.print(hour);
    System.out.print(":");
    System.out.println(minute);
}
```

Exercise 2

The point of this exercise is to practice reading code and to make sure that you understand the flow of execution through a program with multiple methods.

1. What is the output of the following program? Be precise about where there are spaces and where there are newlines.
    a. HINT: Start by describing in words what ping and baffle do when they are invoked.

2. Draw a stack diagram that shows the state of the program the first time ping is invoked.

```
  public static void zoop() {
    baffle();
    System.out.print("You wugga ");
    baffle();
  }

  public static void main(String[] args) {
    System.out.print("No, I ");
    zoop();
    System.out.print("I ");
    baffle();
  }

  public static void baffle() {
    System.out.print("wug");
    ping();
  }

  public static void ping() {
    System.out.println(".");
  }
```

Chapter 4 Conditionals and recursion

Exercise 1

This exercise reviews the flow of execution through a program with multiple methods. Read the following code and answer the questions below.

```
public class Buzz {
    public static void baffle(String blimp) {
        System.out.println(blimp);
        zippo("ping", -5);
    }

    public static void zippo(String quince, int flag) {
        if (flag < 0) {
            System.out.println(quince + " zoop");
        } else {
            System.out.println("ik");
            baffle(quince);
            System.out.println("boo-wa-ha-ha");
        }
    }

    public static void main(String[] args) {
        zippo("rattle", 13);
    }
}
```

1. Write the number 1 next to the first statement of this program that will be executed. Be careful to distinguish things that are statements from things that are not.
2. Write the number 2 next to the second statement, and so on until the end of the program. If a statement is executed more than once, it might end up with more than one number next to it.
3. What is the value of the parameter blimp when baffle gets invoked?
4. What is the output of this program?

Exercise 2

What is the output of the following program?

```
public class Narf {

    public static void zoop(String fred, int bob) {
        System.out.println(fred);
        if (bob == 5) {
            ping("not ");
        } else {
            System.out.println("!");
        }
    }

    public static void main(String[] args) {
        int bizz = 5;
        int buzz = 2;
        zoop("just for", bizz);
        clink(2*buzz);
    }

    public static void clink(int fork) {
        System.out.print("It's ");
        zoop("breakfast ", fork) ;
    }

    public static void ping(String strangStrung) {
        System.out.println("any " + strangStrung + "more ");
    }
}
```

Chapter 5 GridWorld: Part 1

Exercise 1

Chapter 6 Value methods

Exercise 1

Write a method named isDivisible that takes two integers, n and m and that returns true if n is divisible by m and false otherwise.

Exercise 2

If you are given three sticks, you may or may not be able to arrange them in a triangle. For example, if one of the sticks is 12 inches long and the other two are one inch long, you will not be able to get the short sticks to meet in the middle. For any three lengths, there is a simple test to see if it is possible to form a triangle:
"If any of the three lengths is greater than the sum of the other two, then you cannot form a triangle. Otherwise, you can."

Write a method named isTriangle that it takes three integers as arguments, and that returns either true or false, depending on whether you can or cannot form a triangle from sticks with the given lengths.

The point of this exercise is to use conditional statements to write a value method.

Chapter 7 Iteration and loops

Exercise 1

Consider the following code:

```
public static void main(String[] args) {
    loop(10);
}

public static void loop(int n) {
    int i = n;
    while (i > 0) {
        System.out.println(i);
        if (i%2 == 0) {
            i = i/2;
        } else {
            i = i+1;
        }
    }
}
```

1. Draw a table that shows the value of the variables i and n during the execution of loop. The table should contain one column for each variable and one line for each iteration.
2. What is the output of this program?

Exercise 2

Let's say you are given a number, a, and you want to find its
square root. One way to do that is to start with a very rough guess about
the answer, x0, and then improve the guess using the following formula:

x1 = (x0 + a/x0)/2

For example, if we want to find the square root of 9, and we start with x0 = 6,
then x1 = (6 + 9/6)/2 = 15/4 = 3:75, which is closer.
We can repeat the procedure, using x1 to calculate x2, and so on. In this
case, x2 = 3:075 and x3 = 3:00091. So that is converging very quickly on the
right answer(which is 3).

Write a method called squareRoot that takes a double as a parameter and
that returns an approximation of the square root of the parameter, using this
technique. You may not use Math.sqrt.

As your initial guess, you should use a/2. Your method should iterate until
it gets two consecutive estimates that differ by less than 0.0001; in other
words, until the absolute value of $x_n - x_{n-1}$ is less than 0.0001. You can use
Math.abs to calculate the absolute value.

Chapter 8 Strings and things

Exercise 1

Write a method that takes a String as an argument and that prints the letters backwards all on one line.

Exercise 2

```
public class BadString {
    public static void main(String[] args) {
        processWord("banana");
    }

    public static void processWord(String s) {
        char c = getLastLetter(s);
        System.out.println(c);
    }

    public static char getLastLetter(String s) {
        int index = s.length(); // WRONG!
        char c = s.charAt(index);
        return c;
    }
}
```

```
Exception in thread "main" java.lang.StringIndexOutOfBoundsException:
String index out of range: 6
      at java.lang.String.charAt(String.java:694)
      at BadString.getLastLetter(BadString.java:24)
      at BadString.processWord(BadString.java:18)
      at BadString.main(BadString.java:14)
```

Read the stack trace in Section 8.4 and answer these questions:

- What kind of Exception occurred, and what package is it defined in?
- What is the value of the index that caused the exception?
- What method threw the exception, and where is that method defined?
- What method invoked charAt?
- In BadString.java, what is the line number where charAt was invoked?

Chapter 9 Mutable objects

Exercise 1

1. For the following program, draw a stack diagram showing the local variables and parameters of main and riddle, and show any objects those variables refer to.
2. What is the output of this program?

```
public static void main(String[] args)
{
    int x = 5;
    Point blank = new Point(1, 2);

    System.out.println(riddle(x, blank));
    System.out.println(x);
    System.out.println(blank.x);
    System.out.println(blank.y);
}

public static int riddle(int x, Point p)
{
    x = x + 7;
    return x + p.x + p.y;
}
```

The point of this exercise is to make sure you understand the mechanism for passing Objects as parameters.

Exercise 2

1. For the following program, draw a stack diagram showing the state of the program just before distance returns. Include all variables and parameters and the objects those variables refer to.
2. What is the output of this program?

```
public static double distance(Point p1, Point p2) {
    int dx = p1.x - p2.x;
    int dy = p1.y - p2.y;
    return Math.sqrt(dx*dx + dy*dy);
}

public static Point findCenter(Rectangle box) {
    int x = box.x + box.width/2;
    int y = box.y + box.height/2;
    return new Point(x, y);
}

public static void main(String[] args) {
    Point blank = new Point(5, 8);

    Rectangle rect = new Rectangle(0, 2, 4, 4);
    Point center = findCenter(rect);

    double dist = distance(center, blank);

    System.out.println(dist);
}
```

Chapter 10 GridWorld: Part 2

Exercise 1

Chapter 11 Create your own objects

Exercise 1

In the board game Scrabble2, each tile contains a letter, which is used to spell words, and a score, which is used to determine the value of words.

1. Write a definition for a class named Tile that represents Scrabble tiles. The instance variables should be a character named letter and an integer named value.
2. Write a constructor that takes parameters named letter and value and initializes the instance variables.
3. Write a method named printTile that takes a Tile object as a parameter and prints the instance variables in a reader-friendly format.

4. Write a method named testTile that creates a Tile object with the letter Z and the value 10, and then uses printTile to print the state of the object.

The point of this exercise is to practice the mechanical part of creating a new class definition and code that tests it.

Exercise 2

Write a class definition for Date, an object type that contains three integers, year, month and day. This class should provide two constructors. The first should take no parameters. The second should take parameters named year, month and day, and use them to initialize the instance variables.

Write a main method that creates a new Date object named birthday. The new object should contain your birthdate. You can use either constructor.

Chapter 12 Arrays

Exercise 1

Write a method called cloneArray that takes an array of integers as a parameter, creates a new array that is the same size, copies the elements from the first array into the new one, and then returns a reference to the new array.

Exercise 2

Write a method named areFactors that takes an integer n and an array of integers, and that returns true if the numbers in the array are all factors of n (which is to say that n is divisible by all of them). HINT: See Exercise 1.

Chapter 13 Arrays of Objects

Exercise 1

In Blackjack the object of the game is to get a collection of cards with a score of 21. The score for a hand is the sum of scores for all cards. The score for an aces is 1, for all face cards is ten, and for all other cards the score is the same as the rank. Example: the hand (Ace, 10, Jack, 3) has a total score of 1 + 10 + 10 + 3 = 24.

Write a method called handScore that takes an array of cards as an argument and that returns the total score.

Chapter 14 Objects of Arrays

Exercise 1

The goal of this exercise is to implement the shuffling and sorting algorithms from this chapter.

1. Download the code from this chapter from http://thinkapjava.com/code/Card2.java and import it into your development environment. I have provided outlines for the methods you will write, so the program should compile. But when it runs it prints messages indicating that the empty methods are not working. When you fill them in correctly, the messages should go away.
2. If you did Exercise 3, you already wrote randomInt. Otherwise, write it now and add code to test it.

3. Write a method called swapCards that takes a deck (array of cards) and two indices, and that switches the cards at those two locations.
   a. HINT: it should switch references, not the contents of the objects. This is faster; also, it correctly handles the case where cards are aliased.
4. Write a method called shuffleDeck that uses the algorithm in Section 14.2. You might want to use the randomInt method from Exercise 3.
5. Write a method called indexLowestCard that uses the compareCard method to find the lowest card in a given range of the deck (from lowIndex to highIndex, including both).
6. Write a method called sortDeck that arranges a deck of cards from lowest to highest.
7. Using the pseudocode in Section 14.6, write the method called merge. Be sure to test it before trying to use it as part of a mergeSort.
8. Write the simple version of mergeSort, the one that divides the deck in half, uses sortDeck to sort the two halves, and uses merge to create a new, fully-sorted deck.
9. Write the fully recursive version of mergeSort. Remember that sortDeck is a modifier and mergeSort is a function, which means that they get invoked differently:

```
sortDeck(deck);              // modifies existing deck
deck = mergeSort(deck);      // replaces old deck with new
```

Chapter 15 Object-oriented programming

Exercise 1

Download http://thinkapjava.com/code/CardSoln2.java and http://thinkapjava.com/code/CardSoln3.java.

CardSoln2.java contains solutions to the exercises in the previous chapter. It uses only class methods (except the constructors).

CardSoln3.java contains the same program, but most of the methods are object methods. I left merge unchanged because I think it is more readable as a class method.

Transform merge into an object method, and change mergeSort accordingly. Which version of merge do you prefer?

Exercise 2

Transform the following class method into an object method.

```
public static double abs(Complex c) {
    return Math.sqrt(c.real * c.real + c.imag * c.imag);
}
```

Exercise 3

Transform the following object method into a class method.

```
public boolean equals(Complex b) {
    return(real == b.real && imag == b.imag);
}
```

Chapter 16 GridWorld: Part 3

Exercise 1